

Onboard Planning and Scheduling Autonomy within the Scope of the FireBird Mission

Christoph Lenzen* Maria Theresia Wörle* Tobias Göttfert* Falk Mrowka †

Martin Wickler ‡

*Deutsches Zentrum für Luft- und Raumfahrt e. V., German Aerospace Center
Münchener Straße 20, 82234 Weßling, Germany*

For most low orbiting earth observation satellite missions, the timeline is generated on-ground and during dedicated uplink sessions the corresponding tele-commands are sent to the spacecraft. Benefits of this approach are easy maintainability of the complex planning software and quick response times to customer input. However this approach has two major drawbacks: On the one hand the spacecraft behavior is not completely predictable in terms of constraining resources, which means that even detailed modeling requires margins for the on-board resources within the on-ground scheduling algorithms. On the other hand, the reaction time to onboard detected events includes at least the two upcoming ground station contacts, since data downlink and evaluation, (re-)planning and tele-command uplink have to be awaited before the spacecraft can perform new activities. This paper describes the final design and use cases of *VAMOS*, an experiment of *DLR/GSOC*, which will be part of the *FireBird* mission. *VAMOS* consists of a combined onboard / on-ground planning system, which resolves the above mentioned drawbacks by supplying limited onboard autonomy to the satellite, retaining the benefits of a ground based planning system as far as possible.

I. Introduction

For more than twelve years, the *BIRD* mission has been providing infrared image data. Initially planned as follow-up mission, *DLR* (Deutsches Zentrum für Luft- und Raumfahrt e. V., German Aerospace Center) has launched the *FireBird* mission.¹ However this mission does not only supply two successors to the *BIRD* satellite, but both satellites also serve as platforms for multiple highly interesting technological experiments. Especially the second satellite, *Biros* (Berlin Infra-Red Optical System), which will be launched in 2015, supplies features, which make it interesting from the planning point of view:

1. The main payload, the infrared camera, is capable of detecting fire events.
2. The experiment *VIMOS* supplies further image analysis, in particular it allows identifying cloud covered pictures, which may be deleted.²
3. An experimental *OrbComm* modem has been integrated. The *OrbComm* satellite network supplies a global short message service for ground-to-ground communication. This experiment uses *OrbComm* to send short messages from ground to the satellite and vice versa.
4. A loss-less compression of the image data requires margins within the ground planning system.
5. The payload processing unit supports a comparatively high-level application interface based upon the real-time operating system *RODOS*,³ including support of sub-schedules.

*Mission Planning System Engineer, Mission Operations Department, German Space Operations Center

†Head of Mission Planning Team, Mission Operations Department, German Space Operations Center

‡Deputy Head of Mission Operations Department, Mission Operations Department, German Space Operations Center

The baseline for *Biros* operations is to use the existing planning system of the first satellite of the *FireBird* mission, *TET-1*. However the *TET-1* planning system is a ground based planning system, therefore it has two drawbacks which most planning systems for low earth orbiting satellites share:

1. some onboard resources cannot be predicted accurately, thus margins need to be introduced
2. reaction time to onboard detected events includes at least the two upcoming ground station contacts, since data downlink and evaluation, (re-)planning and tele-command uplink have to be awaited before the spacecraft can perform new activities.

To overcome these drawbacks, the mission planning team of *DLR/GSOC* has proposed another experiment for the *Biros* satellite, called *VAMOS* (Verification of Autonomous Mission planning Onboard a Spacecraft). *VAMOS* is an integrated planning system consisting of an onboard component, which is implemented on the payload processing unit (PPU) of *Biros*, and an on-ground component, which extends the *FireBird* mission planning system.

The improvements provided by *VAMOS* are:

i Real time telemetry checks

Real-time telemetry checks are used for deciding whether an activity may be performed or not.

On the *Biros* satellite, the following scenarios are possible:

- (a) The unpredictable state of the memory is checked shortly before execution of each timeline fragment. This way more acquisitions may be executed, because propagation margins are reduced significantly.
- (b) *VAMOS* may trigger a file deletion when detecting a valueless, since cloud covered, image. Again the telemetry check can indicate that an additional acquisition may be activated, which the ground scheduler would have discarded.

ii Onboard event-driven acquisition generation

Onboard triggered events may cause new acquisitions to be generated, together with the corresponding telemetry checks. On the *Biros* satellite, the following scenarios are possible:

- (a) On-ground, a customer identifies a high priority acquisition request, e.g. in order to monitor a volcanic eruption. In case the first opportunity of this request lies before the upcoming uplink station contact, the *OrbComm* modem can be used to generate an onboard event containing the information about when to execute the new request.
- (b) Onboard the infrared camera identifies and triggers a fire event. *VAMOS* may immediately generate a new acquisition request of the region of interest with a higher resolution and activate it during the same pass with a backward looking angle. This scenario of course requires high calculation performance of the image processing unit on board the satellite and possibly the use of the high-torque wheels, a further experiment on *Biros*, which provides very fast slew maneuvers of the satellite.

Although the onboard component of *VAMOS* makes its decisions to activate a request just in time, it does not restrict to a first-come-first-serve approach. Instead, for all ground commanded activities, *VAMOS* supports a standard priority concept such as the one used for the nominal *FireBird* mission or the one used for the TerraSAR-X mission.⁴ For onboard generated acquisition requests however we restrict to one constant priority in order to keep onboard software complexity small.

VAMOS is a continuation of previous work,⁵ where onboard file deletion, based upon cloud detection, should enable the *BIRD* satellite to include additional acquisitions. In contrast to the predecessor's approach, *VAMOS* also allows reacting to onboard detected events. Unfortunately no onboard mechanisms have been implemented on the *BIRD* satellite. *VAMOS* on the other hand has passed the unit tests and is currently being integrated into the *Biros* satellite. Execution of this experiment is planned during the routine phase of the *Biros* satellite, which is expected to end in 2016. In case of success, *VAMOS* may be taken over into the routine operations of *FireBird* in order to increase the scientific output of the mission.

The principles of the design of *VAMOS* have already been described,^{6,7} where focus has been given to the calculation logic. In the following, we also answer the question of how to balance in between the onboard and on-ground functionality, i.e. why we chose this design and how we exploit the capabilities of *Biros*.

II. The FireBird Mission

As continuation of the *BIRD* mission, *FireBird*'s main mission goal is fire observation.¹ In contrast to *BIRD*, the *FireBird* mission will consist of a constellation of two satellites. Its first satellite, *TET-1*, has been launched on 22nd of July 2012. However, *TET-1* has not been launched exclusively for the *FireBird* mission. It has been equipped with several experimental payloads, which have been tested during the on-orbit verification phase. End of 2013 the satellite has been handed over to the *FireBird* mission, involving a complete change in concept with respect to mission planning. Whereas the spacecraft activities during the on-orbit verification phase were planned well in advance and the mission planning tasks were restricted to performing timeline corrections and the command export,⁸ the *FireBird* mission planning system has the typical requirements of an earth observing mission. In particular, an interactive order interface has been requested, including a swath preview functionality, which allows searching for the acquisition opportunities of a specific area of interest, as well as a daily planning run schedule in order to allow flexible usage of the spacecraft. In 2015, the second satellite *Biros* will be launched. In contrast to the experiments on board of *TET-1*, some experiments on board of *Biros* facilitate new mission planning approaches which may increase the benefit for scientific return of an earth observing satellite. Among others, *Biros* includes the following experiments:

1. an *OrbComm* modem, which allows sending short messages via the *OrbComm* satellite network
2. so-called *High Torque Wheels* which allow very fast slew maneuvers
3. *OSIRIS*: a laser communication terminal which is intended to communicate with the ground
4. *VIMOS*: an image analyzing software, which aims to detect floods, bridges, clouds, etc.²
5. *VAMOS*: the onboard scheduling experiment described in this paper

Both satellites base on successor buses of *BIRD* and carry a camera system with a bi-spectral infrared hot spot recognition sensor system and a three-channel optical sensor as multi-functional camera as their main instrument for the *FireBird* mission. Compared to *BIRD*, the infrared images are complemented with optical images, furthermore the constellation of two satellites will bring a reduction of target revisit times. Additionally the *FireBird* satellites are capable of onboard data processing of fire products which may be used

1. to avoid dumping the full images,
2. to trigger onboard events according to fire events and
3. to immediately send the fire events via the *OrbComm* modem to the ground.

III. Balancing of Onboard and On-Ground Features

III.A. Criteria to Consider

In spacecraft operations a common approach to move planning capabilities to the satellite is the concept of goals⁹ as applied within the mars rover planning system *OSIRIS* of NASA.¹⁰ This means that the spacecraft is in charge of selecting a feasible set of goals and translating it into a valid timeline. It may also generate new goals according to onboard detected events, which may be provided e.g. by an image analyzing software. Whenever new goals have been defined, the spacecraft may select a better set of goals and translate it into new commands, discarding the ones of the preceding planning run. The benefits of this approach are:

1. An immediate response to onboard detected events may generate goals of high importance and timeliness, which makes them extremely valuable.
2. The continuous observation of real time telemetry allows including goals which wouldn't be planned on ground according to conservative estimations. This allows increasing the utilization of the spacecraft, but it usually only adds goals of minor value, since the high-priority goals would have been part of a ground-propagated timeline, too.

However this approach implies drawbacks, too:

1. major implementation effort for onboard software
2. reduced control of the spacecraft behavior implies reduced feedback reliability for the scientist resp. customer

At *GSOC* (German Space Operations Center – DLR Space Operations), our experiences have shown that the customer usually wants to know what will happen and why it happened. He will accept a simple rule such as *an acquisition request with greater priority has displaced my order*; however he will not appreciate a timeline optimization according to updated resource states, which may on the one hand include several previously unscheduled requests, but on the other hand may exclude as many previously scheduled ones. Instead the desired approach is to maintain a stable timeline, which – in urgent cases – may be disturbed, preferably with confirmation about the deletion of a previously scheduled request, but in any case with immediate notification about the deletion in order to allow the ingestion of an alternative order.¹¹

Additionally, implementing complex onboard software is much more expensive than implementing similar software to be executed on-ground and calculation power is by far greater when using servers on-ground than using an onboard unit of a spacecraft. On *Biros*, the *VAMOS* software is integrated into the PPU's operating system *RODOS*.³ It assures real-time execution of *VAMOS*' processing cycle, provided the software does not exceed its limited calculation budget. Software development for *RODOS* is done in C++, which is a good choice when code performance is important, however modern languages allow faster development of equally reliable code, in particular when sufficient calculation power is available. As part of the PPU software, *VAMOS* can only be updated as part of a complete PPU software upload. These circumstances confirm our decision to keep as much code complexity on-ground as possible.

All in all, these circumstances confirm our decision to keep as much code complexity on-ground as possible.

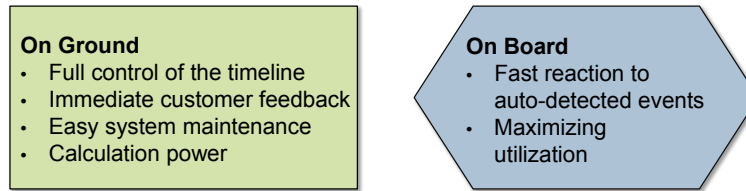


Figure 1. Trade-off in between onboard and on-ground calculation.

In the following section, we discuss the high-level design of *VAMOS*, which has been derived from the considerations of this chapter.

III.B. Design of VAMOS

As implied above, we prefer the main scheduling process to remain on-ground. On the one hand, this allows easier maintainability of the complex scheduling software as well as a double check of on-ground generated commands. On the other hand this allows using the incremental planning system,¹¹ supplying full control over the on-ground generated timeline. As additional benefit, the planning system may start with deactivated onboard autonomy, resulting in a well-known standard planning system, which allows step-by-step integration of the new features into an existing, proven system.

The first step of onboard autonomy addresses the inaccurate on-ground resource propagation. The onboard timeline selection (*OBoTiS*) enables the satellite to activate and deactivate parts of the onboard timeline according to real-time onboard telemetry. So in the beginning, the timeline will be generated completely on-ground.

The second step of onboard autonomy will be the generation of onboard event triggered timeline extension (*OBETTE*), i.e. the generation of further commands according to onboard detected events. The proper integration of these timeline extensions into the existing timeline will be assured by *OBoTiS*, which is not only in charge of checking the resources, but also of checking no-overlap constraints of all subsections of the timeline.

III.C. OBoTiS

The planning cycle in our integrated system begins with the on-ground scheduler. It generates a timeline, consisting of multiple timeline fragments, which can be activated resp. deactivated individually. Usually each timeline fragment corresponds to a planning request. Each timeline fragment is given the information, which time interval it allocates. *OBoTiS*, the onboard counterpart of the on-ground scheduler, is in charge of never activating two timeline fragments whose time intervals overlap. Additionally, the on-ground scheduler adds information to each timeline fragment about onboard resource bounds, which indicate whether the timeline fragment may be activated. *OBoTiS* checks these bounds against the real-time telemetry of the satellite and decides just in time whether the timeline fragment shall be executed or not.

Note that this check does not compare the consumption of the timeline extension against the availability, but it merely checks the current value of the telemetry against the bound of the timeline fragment. This way the on-ground scheduler may include further consumptions into this bound, namely all those of future timeline fragments which have greater priority and therefore must not be blocked by the timeline fragment which is about to be considered.

Without loss of generality we consider all resource bounds to be upper bounds to the corresponding onboard telemetry value. For an example, one may think of the onboard mass memory, which is populated using a loss-less compression, which implies an unknown compression ratio, see fig. 2.

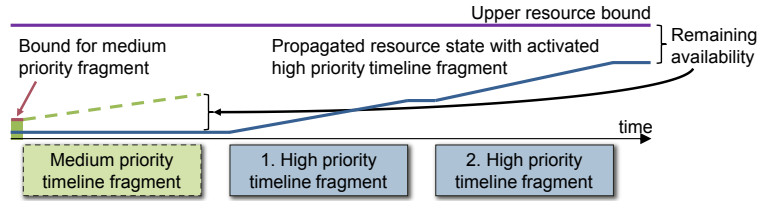


Figure 2. Calculation of a resource bound for a medium-priority timeline fragment.

Although this approach avoids blocking of high-priority timeline fragments by earlier timeline fragments of smaller priority due to resource conflicts, it can not avoid blocking of high priority timeline fragments due to time interval overlaps. Therefore the onboard component must check overlapping timeline fragments in descending order of priority. The execution time of these checks however may be calculated on-ground as part of the resource bound check, see fig. 3. More details on the calculations can be found in preceding papers on *VAMOS*.^{6,7}

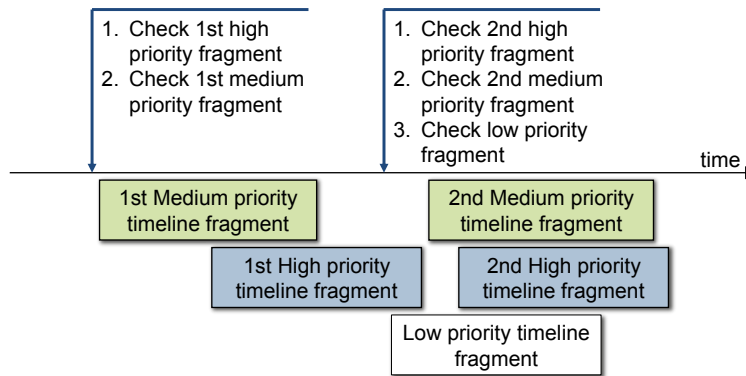


Figure 3. Check of overlapping timeline fragments in descending order of priority.

One last refinement of this approach remains within the on-ground scheduler: In case there exist high priority timeline fragments, which consume considerably more resources than other standard timeline fragments, one may define lower bound checks on standard timeline fragments. Whereas the upper bound on a standard timeline fragment assures that succeeding high priority timeline fragments won't be blocked by the consumption of the standard timeline fragment, the lower bound indicates that none of the succeeding high priority timeline fragments may be scheduled anyway in case this lower bound is obeyed. This lower bound needs to be combined with an upper bound, which may omit the high priority timeline fragments. Together, this pair of upper and lower bound is an alternative pass criterion to the standard upper bound check. It

allows the current timeline fragment to be executed, because the execution of succeeding higher priority timeline fragments is known to be impossible anyway, see fig. 4. In case there exist more than one future timeline fragment with greater priority, the lower bound needs to be set to the maximum of all individually calculated lower bounds in order to assure that this timeline fragment does not block any of the future high priority timeline fragments. On *Biros*, we haven't implemented this alternative check, since it implies further onboard complexity for a non-essential optimization of the experiment.

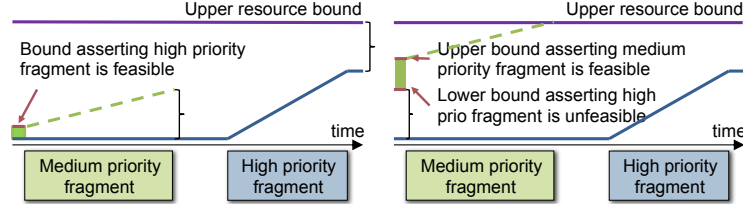


Figure 4. Alternative check with lower bound allows executing earlier medium priority timeline fragments knowing that a succeeding high-priority timeline fragment won't pass its check anyway.

An interesting point of this approach is that the bound calculated for one timeline fragment and one resource is independent from the history, i.e. the previously executed timeline fragments: when scheduling a timeline fragment at some time, the resource bound only depends on the upper resource bound, the consumptions of the timeline fragments which are scheduled later in time and the consumption of the timeline fragment itself. This corresponds to the intuition that deciding what to do next only requires the knowledge of the current state and the requested timeline fragments of the future, see fig. 5.

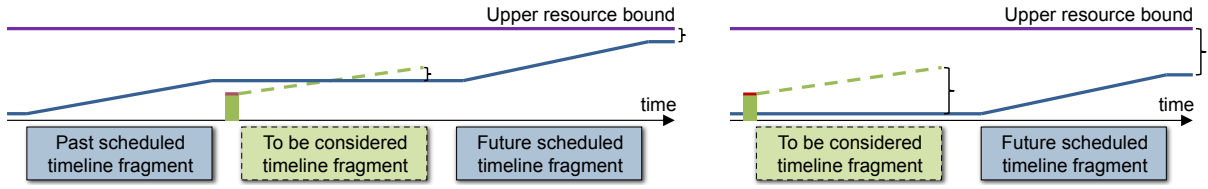


Figure 5. The calculated bound of the timeline fragment is independent from the history.

For the on-ground scheduler, this implies a simple greedy algorithm, adding all timeline fragments descending in priority and calculating the resource bound of each timeline fragment on basis of the currently propagated resource profile when adding the fragment to the timeline. The only challenge in this approach are conflicting timeline fragment combinations: since one knows that only one of multiple conflicting timeline fragments may become activated, one may restrict to the maximum consumption of the conflicting combinations. This however may imply a major performance issue because the number of conflicting combinations can increase exponentially with the number of considered timeline fragments, especially if not only time interval conflicts but also resource conflicts using optimistic calculation are considered.

On *Biros* we will start blocking lower-priority timeline fragments which overlap with timeline fragments of greater priority and we will not detect combinations which are in conflict with each other with respect to optimistic resource calculation. However as this is part of the on-ground scheduler, an upgrade may be implemented any time without the need of updating the onboard software.

III.D. OBETTE

When *OBoTiS* has proven its functionality, the second step in automation will be activated, the generation of new command sequences, i.e. timeline fragments, onboard the spacecraft. Obviously this command generation is particularly critical as new commands are autonomously ingested into an existing timeline, without further checks, which a ground-based system might include. Fortunately we can use *OBoTiS* for integration into the existing timeline. All we need to do is

1. define the time interval which the new timeline fragment allocates,
2. derive the *time of activation check* for the new timeline fragment,
3. calculate the resource bounds for the new timeline fragment and

4. generate the commands for the new timeline fragment.

In order to assure proper command generation, we use a procedure-like approach, i.e. we store a template of several commands onboard, which is copied and filled with parameters when a new timeline fragment is generated. One of these parameters will be the execution time, from which the time interval of the timeline fragment is derived, which is the first step *OBETTE* has to perform.

Thereafter the *time of activation check* will be calculated. For this, *OBETTE* compares the time interval of the new timeline fragment with the time intervals of all existing overlapping timeline fragments:

1. Let *maxCheck* denote the earliest *time of activation check* of all existing timeline fragments which have an overlapping time interval and which have smaller priority than the new timeline fragment. Either all of these existing timeline fragments need to be deleted or the time of activation check of the new timeline extension must be chosen earlier than *maxCheck*.
2. Let *minCheck* denote the latest time of activation check of all existing timeline fragments which have an overlapping time interval and which have greater priority than the new timeline fragment. Either the new timeline fragment must be discarded or the time of activation check of the new timeline extension must be chosen later than *minCheck*. In case one of the existing timeline fragments of greater priority starts later than the new timeline fragment, one has no choice but to discard the new timeline fragment.

Next step in preparation of the new timeline fragment is the generation of the resource bounds. With the capabilities of an on-ground system, one might think of including a resource propagator, which calculates the exact bound at activation time, which asserts that activating this timeline fragment does not block any timeline fragment of greater priority. However this would imply major implementation effort for onboard software and the result would require considerable onboard calculation power. We therefore chose the following, simplified approach:

Assumptions:

1. The resource propagation restricts to an affine linear model, optionally with lost values calculation¹²
2. The resource propagation is *independent from the value*, in particular the consumption of a timeline fragment on a modeled resource is independent from the current state of the resource. More precise:

Let $f(t)$ denote the propagated value of a resource at time t , including the accumulated values which have not been consumed, because an upper limit has been reached. For any two times $t_0 < t_1$: $f(t_1) - f(t_0)$ is independent of $f(t_0)$.

3. All onboard generated timeline fragments are given the same priority.

An example, which needs to be approximated, because it violates 2: The state of charge of a battery increases less the closer the state of charge reaches its maximum. An approximation in such a situation, which fulfills the assumption, may be to use only 80% of the battery's capacity and use the charge rate at *state of charge* = 80% as worst case approximation.

In this situation, the on-ground scheduler may calculate remaining availability profiles, which indicate for each time t , how much resource is available provided that all timeline fragments of greater priority and *time interval* $> t$ must be given sufficient resources. This profile is transmitted to the satellite, together with the profile of the on-ground scheduler's resource propagation and – of course – the timeline fragments. From these two profiles, *OBETTE* can derive the proper upper bound for each resource as shown in fig. 6. Again it turns out that the history is canceled within the calculation of the telemetry threshold. More details can be found in preceding papers on *VAMOS*.^{6,7}

When all preceding steps have been performed, all required actions are taken such that *OBOTIS* will only activate the new timeline fragment in case

- it does not conflict with an already activated timeline fragment,
- just-in-time telemetry checks show that sufficient resources are available and

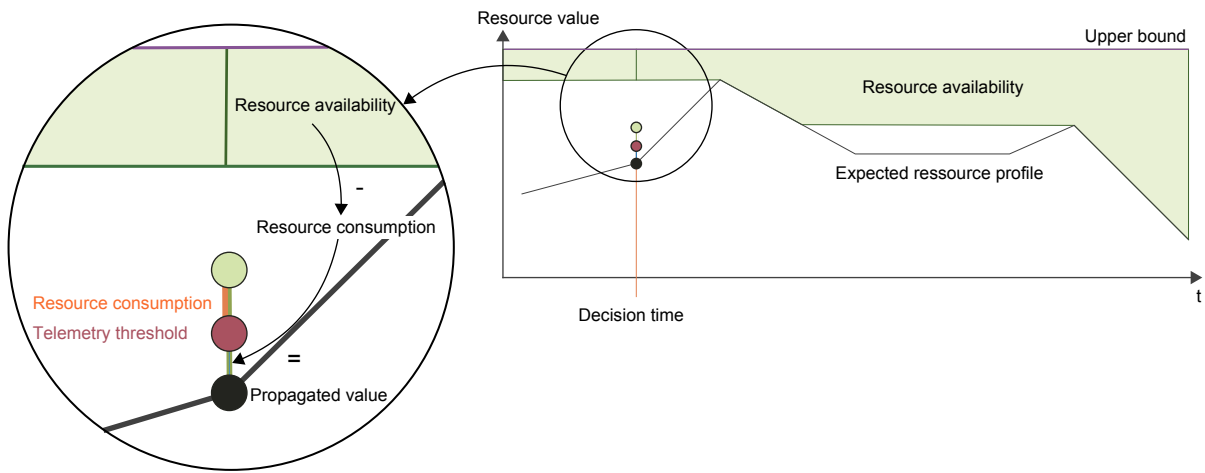


Figure 6. Calculating the resource bounds for onboard generated timeline fractions.

- activation of the new timeline fragment will not cause a higher-priority timeline fragment being rejected later on.

Therefore *OBETTE* now can generate the commands by copying a predefined command template and filling in the parameters which are supplied by the triggering event. From now on, the above described *OBOTIS* mechanism decides whether the newly generated timeline fragment will be executed or discarded in the same way as it does for all on-ground generated timeline fragments.

III.D.1. *OBETTE* Margins

Although onboard generation of timeline fragments will not disturb existing timeline fragments of greater priority, the other way round is not guaranteed. For example, let a high priority event generate a new timeline fragment for execution one hour in the future. The resource bound checks within the next hour are still those calculated on-ground, without knowledge of this additional high priority timeline fragment. Therefore it may happen that a low priority timeline fragment is executed, consuming the resource which would have been required for activating the high priority timeline fragment later on, see fig. 7.

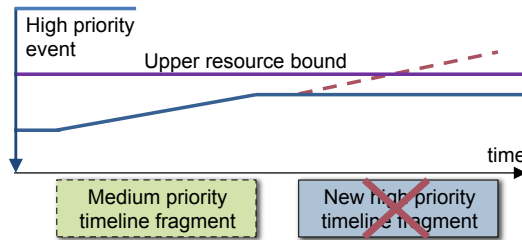


Figure 7. Missing resource margin causes an onboard generated high-priority timeline fragment to be rejected.

To solve this issue, we recommend to reduce the on-ground calculated resource bound of all timeline fragments of lower priority by the consumption of one or two onboard generated timeline fragments. This amount should be sufficient, because the margin will be restored automatically after executing an onboard generated timeline fragment.

This solution also covers the case of a low-priority timeline fragment lying in the past. In case there is not sufficient margin, the low priority timeline fragment would be rejected in order to assure sufficient resources for a high priority event at any time.

III.D.2. *Multiple On-Board Priorities*

OBETTE's approach of generating timeline fragments can be extended to using multiple priorities of events. Since the remaining availability profile of a given event priority must reflect all future consumptions of timeline

fragments of greater priority, one remaining availability profile for each event priority must be calculated and transmitted to the satellite. However in this case, onboard generated timeline fragments of different priorities may corrupt the priority concept, because no update of the remaining availability profile of the lower event priority is foreseen in case an onboard generated timeline fragment of greater priority is generated for some future time.

Again we consider the example of a high-priority event generating a new timeline fragment for execution one hour later. The resource consumptions of this timeline fragment are not reflected in the remaining availability profiles for low-priority events. Therefore a low-priority event that occurs shortly afterwards and generates a timeline fragment with execution time earlier than the one of the high-priority timeline fragment, will generate resource bounds which do not reflect the consumptions of the high-priority timeline fragment. This means that the low-priority timeline fragment may be executed at the expense of the high-priority timeline fragment.

IV. Use Cases of VAMOS within the FireBird Mission

IV.A. Increased Resource Usage

Obviously the first use case will be to allow *OBoTiS* activating as many timeline fragments as possible, according to the resource bound and time interval checks. The most interesting resource with respect to telemetry checks will be the memory resource: First, the data compression onboard is a loss-less compression with unpredictable compression ratio. Second, when a new image has been written into the memory, *VIMOS* may check the image for cloud coverage. In case the picture is evaluated as valueless, it will be deleted onboard and the onboard memory will be available for other image requests. Note that the memory bound reflects the whole future of the planning horizon, therefore this bound includes the modeling of the downlink capacity.

IV.B. Fire Zoom

As mentioned in the introduction, the main infrared camera is capable of detecting fire events. The following scenario therefore may be used as a motivation:

- The satellite acquires large scanning images with low resolution while looking a little ahead in flight direction and examines these pictures for fire events.
- When a fire event is detected, *OBETTE* generates a timeline fragment for a high-resolution image of the area where the fire has been detected. This timeline fragment will be acquired looking backwards in order to allow execution during the same pass.
- *OBoTiS* executes the new timeline fragment, in case no conflicts exist.

The data of the scanning images can be deleted immediately after evaluation. Only the high-resolution images have to be transmitted to the ground.

For missions whose only purpose is the fire detection it may be sufficient to restrict to this use case. Then one may consider using specialized hardware such as proposed by *ASAP*.¹³ However for missions in which the observation of events shall be integrated into a nominal timeline, the approach used by *VAMOS* should be considered. Nevertheless this use case will not be the primary goal of the *VAMOS* experiment, on the one hand because it is challenging with respect to timing constraints, and on the other hand because we prefer the following:

IV.C. Anytime Commanding via OrbComm

The *OrbComm* modem, which is integrated into the *Biros* satellite, will be capable of sending and receiving short messages. Obviously this is not a connection which can be used for commanding, however it may supply a way to send information snippets, e.g. the time and angle in which *Biros* may observe an interesting scene. For instance, in case some unexpected event like a volcanic eruption is registered on earth, the information about time and looking-angle of the upcoming observation opportunity for *Biros* may be sent to the spacecraft. In the best case, the image may be taken within a few minutes after reception of the request from ground.

In case of a forest fire, the image which may have been triggered by an *OrbComm* message may furthermore be directly evaluated onboard and the resulting summary, e.g. the location of the hot spots, may be sent back to the ground via *OrbComm*, too. This way the whole cycle, consisting of uplink, data acquisition, image analysis and downlink, might be reduced to a few minutes, provided the target is visible.

V. Conclusion and Outlook

The concept of limited onboard autonomy embedded into a ground-based planning system, as presented in this paper, combines the main benefits of onboard and on-ground scheduling. Its onboard component supports immediate reaction to onboard detected events as well as maximum resource exploitation by telemetry regulated timeline execution. The ground-based planning system on the other hand supplies user control as far as desired, possibly up to an extent as provided by the Incremental Planning System.¹¹ The concept therefore shall not restrict to the *Biros* satellite, but will be taken into account for all future missions at *GSOC* for which onboard autonomy shall increase the scientific return. Although the ground-based scheduling system has to implement some features in order to implement the interfaces of the onboard component, the additional effort of the overall system is limited by the restricted complexity of the onboard component, which we consider as the main additional effort.

The main purpose of the experiment *VAMOS* onboard *Biros* is to demonstrate the capabilities of onboard autonomy as described in this paper. The baseline therefore will be to emulate the *OrbComm* modem using a standard uplink station and to ingest further acquisition requests in order to supply an overbooked timeline. Nevertheless in case *OrbComm* works fine – which we all hope – or in case the customers request more acquisitions than the satellite may serve, *VAMOS* may supply a real benefit to the *FireBird* mission. In this case we intend to include *VAMOS* into nominal operations of the *FireBird* mission.

References

- ¹Reile, H., Lorenz, E., and Terzibaschian, T., “The FireBird Mission – A Scientific Mission for Earth Observation and Hot SpotDetection,” *Small Satellites for Earth Observation, Digest of the 9th International Symposium of the International Academy of Astronautics*, Wissenschaft und Technik Verlag, ISBN 978-3-89685-574-9
- ²Götz, K. A. M., Huber, F., and Schönermark, M., “VIMOS - Autonomous Image Analysis on Board of BIROS,” *Automatic Control in Aerospace, Volume # 19 — Part # 1*, Würzburg, Germany, 2013
- ³“RODOS”, URL: <http://www.montenegros.de/sergio/rodos/rodos-de.html> [cited March 27, 2014]
- ⁴Lenzen, C., Wörle, M. T., Mrowka, F., Geyer, M. P., and Klaehn, R., “Automated Scheduling for TerraSAR-X/TanDEM-X,” *IWPSS 2011*, Darmstadt, Germany, June 08-10, 2011
- ⁵Axmann, R. and Wickler, M., “Development and Verification of an Autonomous On-Board Mission Planning System - an Example from the BIRD Satellite,” *SpaceOps 2006*, Rome, Italy, June 06-23, 2006
- ⁶Wörle, M. T., and Lenzen, C., “Ground Assisted Onboard Planning Autonomy with VAMOS,” *International Workshop for Planning and Scheduling in Space*, Moffett Field, CA, USA, March 25-26, 2013
- ⁷Wille, B. and Wörle, M. T., and Lenzen, C., “VAMOS – Verification of Autonomous Mission Planning On-board a Spacecraft,” *19th IFAC Symposium on Automatic Control in Aerospace*, Würzburg, Deutschland, September 02-06, 2013
- ⁸Wörle, M. T., Spörl, A. and Hartung, J., Lenzen, C., Mrowka, F., Wickler, M., and Braun, A., “Mission Planning System for the TET-1 OnOrbitVerification Mission,” *SpaceOps 2014, 13th International Conference on Space Operations*, Pasadena, California, May 5 - 9, 2014
- ⁹Chien, S., Smith, B., Rabideau, G., Muscettola, N., and Rajan, K., “Automated Planning and Scheduling for Goal-Based Autonomous Spacecraft,” *IEEE Intelligent Systems*, vol. 13, no. 5, pp. 50-55, September / October 1998
- ¹⁰Castano, R., Estlin, T., Anderson, R. C., Gaines, D. M., Castano, A., Bornstein, B., Chouinard, C., and Judd, M., “OASIS: Onboard autonomous science investigation system for opportunistic rover science,” *Journal of Field Robotics*, Vol 2007, pp. 379-397
- ¹¹Wörle, M. T., Lenzen, C., Mrowka, F., Göttfert, T., Spörl, A., Grishchkin, B., and Wickler, M., “The Incremental Planning System - GSOC’s Next Generation Mission Planning Framework,” *SpaceOps 2014, 13th International Conference on Space Operations*, Pasadena, California, May 5 - 9, 2014
- ¹²Lenzen, C., Mrowka, F., Spörl, A., and Klaehn, R., “Scheduling Formations and Constellations,” *SpaceOps 2010, 11th International Conference on Space Operations*, Huntsville, Alabama, April 25 - 30, 2010
- ¹³Balagurin, O., Fellingner, G., Garcia, B., and Kayal, H., “ASAP: autonomous dynamic scheduling for small satellites” *64th International Astronautical Congress*, Beijing, China, September 23 - 27, 2013